Co-Sacle Conv-Attentional Image Transformers

CoaT Paper Review

Sangho Kim

Table of Contents

- Introduction
- Combination of CNN and Self-Attention
- Review Self-Attention in ViT
- Methods
- Model Architecture
- Experiments
- Conclusion

Introduction

- In essence, both the convolution and attention operations address the fundamental representation problem for structured data.
- The receptive fields in CNNs are gradually expanded through a series of convolution operations.
- The attention mechanism is different from the convolution operations.
  - 1) The receptive field at each location or token in self-attention readily covers the entire input space since each token is matched with all tokens.
  - 2) The self-attention operation for each pair of tokens computes a dot product between the query and the key to weight the value.
- In the self-attention mechanism, the weights are dynamically computed based on the similarity or affinity between every pair of tokens.
- As a consequence, the self-similarity operation in the self-attention provides modeling means that are potentially more adaptive and general than convolution operations.

Table of Contents

- Introduction
- Combination of CNN and Self-Attention
- Review Self-Attention in ViT
- Methods
- Model Architecture
- Experiments
- Conclusion

Combination of CNN and Self-Attention

- In this paper, the authors propose the ideas combined convolutional networks and self-attention mechanism.
- The one of the advantages in CNNs is capturing local spatial information but it does not have long-range dependencies.
- And the self-attention mechanism has a advantage of long-range dependencies because it computes similarity between every pair of tokens.
- Therefore, we develop Co-scale conv-attentional image Transformers (CoaT).
- The contributions of our work are summarized as follows:
    - 1) We introduce a co-scale mechanism to image Transformers by maintaining encoder branches at sperate scales while engaging attention across scales.
    - 2) We design a conv-attention module to realize relative position embeddings with convolutions that achieves significantly enhanced computation efficiency.

Table of Contents

- Introduction
- Combination of CNN and Self-Attention
- Review Self-Attention in ViT
- Methods
- Model Architecture
- Experiments
- Conclusion

Review Self-Attention in ViT

- Revisit Scaled Dot-Product Attention

- Transformers take as input a sequence of vector representations $X \in \mathbb{R}^{N \times C}$.
- The projection of the whole sequence generate representations $Q, K, V \in \mathbb{R}^{N \times C}$.
- The scaled dot-product attention is formulated as:

$$\mathrm{Att}(X) = \mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{C}}\Big)V$$

Table of Contents

- Introduction
- Combination of CNN and Self-Attention
- Review Self-Attention in ViT
- Methods
- Model Architecture
- Experiments
- Conclusion

Methods

- **Factorized Attention Mechanism**

- For equation of scaled dot-product attention in ViT, it leads to the $O(N^2)$ space complexity and $O(N^2C)$ time complexity.
- Inspired by recent works, we approximate the attention map by factorizing it using two functions $\phi(\cdot), \psi(\cdot): \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times C'}$.
- Here, we develop our factorized attention mechanism following LambdaNets with ϕ as the identity function and ψ as the softmax:

$$\text{FactorAtt}(X) = \frac{Q}{\sqrt{C}}\left(\text{softmax}(K)^\top V\right)$$

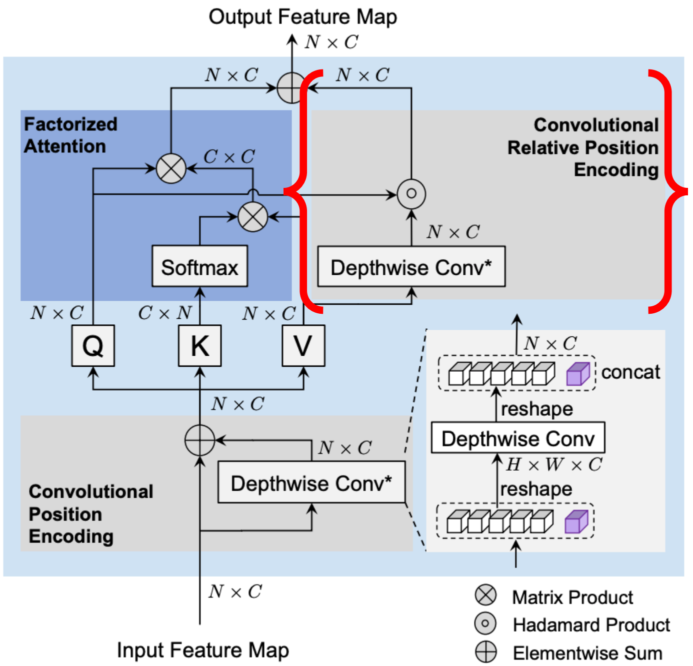
- This factorized attention takes $O(NC + C^2)$ space complexity and $O(NC^2)$ time complexity.

Methods

- **Convolution as Position Encoding**

- Without the position encoding, the Transformer is only composed of linear layers and self-attention modules.
- Thus, the output of a token is dependent on the corresponding input without awareness of any difference in its locally nearby features.
- This property is unfavorable for vision tasks such as semantic segmentation (e.g. the same blue patches in the sky and the sea are segmented as the same category).

Methods



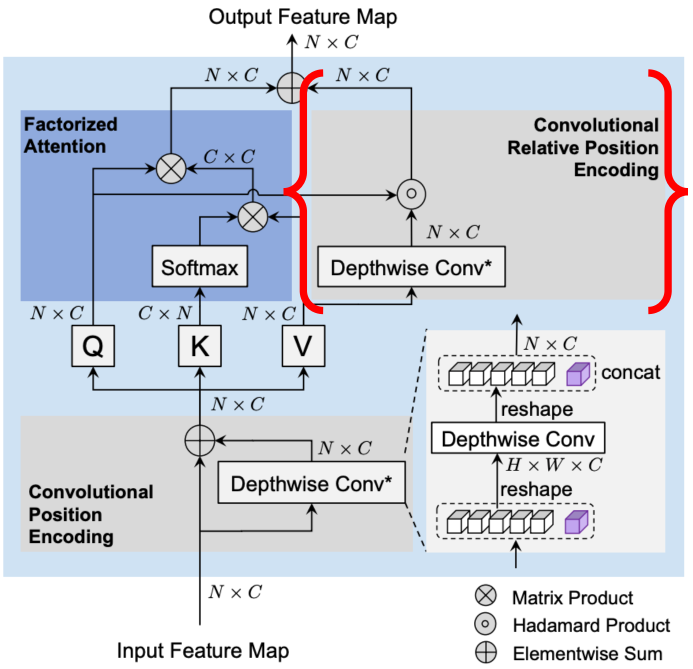
- **Convolution Relative Position Encoding**

- To enable vision tasks, ViT insert absolute position embeddings into the input, which may have limitations in modeling relations between local tokens.
- Instead, we can integrate a relative position encoding $P = \{\mathbb{p}_i \in \mathbb{R}^C, i = -\frac{M-1}{2}, \ldots, \frac{M-1}{2}\}$ with window size M to obtain the relative attention map $EV \in \mathbb{R}^{N\times C}$, if tokens are regarded as a 1D sequence:

$$\mathrm{RelFactorAtt}(X) = \frac{Q}{\sqrt{C}}\left(\mathrm{softmax}(K)^{\top}V\right) + EV$$

- Where the encoding matrix $E \in \mathbb{R}^{N\times N}$ has elements $E_{ij} = 1(i,j)\boldsymbol{q}_i \cdot \boldsymbol{p}_{j-1}, 1 \leq i,j \leq N$ in which $1(i,j)$ is an indicator function.

Methods



- **Convolution Relative Position Encoding**

- Unfortunately, the EV term in above function still requires $O(N^2)$ space complexity and $O(N^2C)$ time complexity.
- In CoaT, we propose to simplify the EV term to $\widehat{EV}$ by considering each channel in the query, position encoding and value vectors as internal heads.

$$E_{ij}^{(l)} = \mathbb{1}(i,j) q_i^{(l)} p_{j-i}^{(l)}, \quad \hat{EV}_i^{(l)} = \sum_j E_{ij}^{(l)} v_j^{(l)}$$

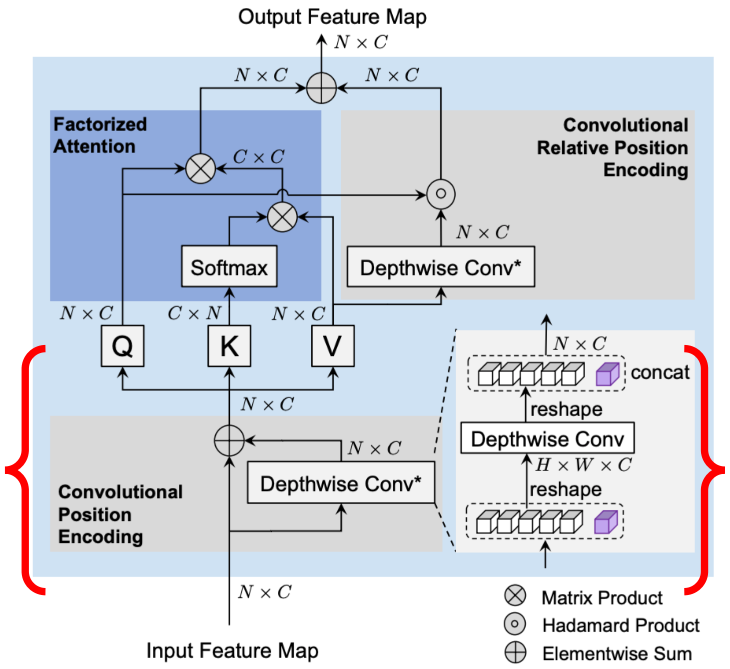
- In practice, we can use a 1D depthwise convolution to compute $\widehat{EV}$:

$$\hat{EV}^{(l)} = Q^{(l)} \circ \mathrm{Conv1D}(P^{(l)}, V^{(l)}),$$
$$\hat{EV} = Q \circ \mathrm{DepthwiseConv1D}(P, V)$$

$$\hat{EV}^{\mathrm{img}} = Q^{\mathrm{img}} \circ \mathrm{DepthwiseConv2D}(P, V^{\mathrm{img}})$$
$$\hat{EV} = \mathrm{concat}(\hat{EV}^{\mathrm{img}}, \mathbf{0})$$
$$\mathrm{ConvAtt}(X) = \frac{Q}{\sqrt{C}}\left(\mathrm{softmax}(K)^\top V\right) + \hat{EV}$$

- This attention computes only $O(NC)$ space complexity and $O(NCM^2)$ time complexity, aiming to achieve better efficiency.

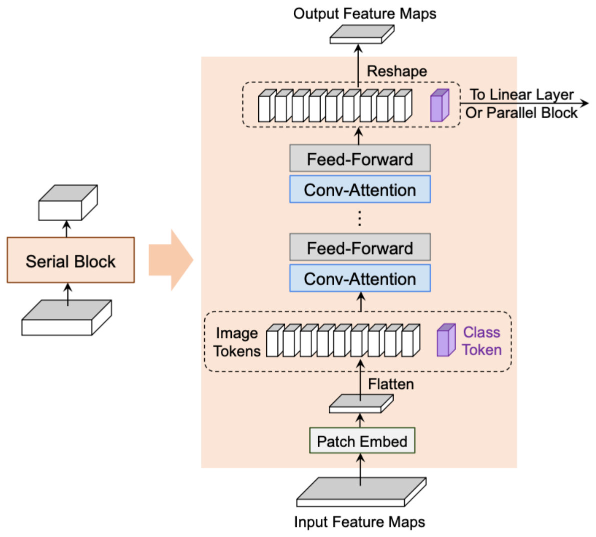
Methods



- **Convolution Position Encoding**

- We extend the idea of convolutional relative position encoding (CRPE) to a general convolutional position encoding (CPE) case.
- CRPE models local position-based relationships between queries and values.
- Similar to the absolute position encoding, we would like to insert the position relationship into the input images features directly to enrich the effects of relative position encoding.
- We insert a depthwise convolution into the input features X and concatenate the resulting position-aware features back to the input features.
- We set kernel size to 3 for the convolutional position encoding.
- We set kernel size to 3, 5 and 7 for image features from different attention heads for convolutional relative position encoding.
- Our work focuses on applying convolution as relative position encoding and a general position encoding with the factorized attention.

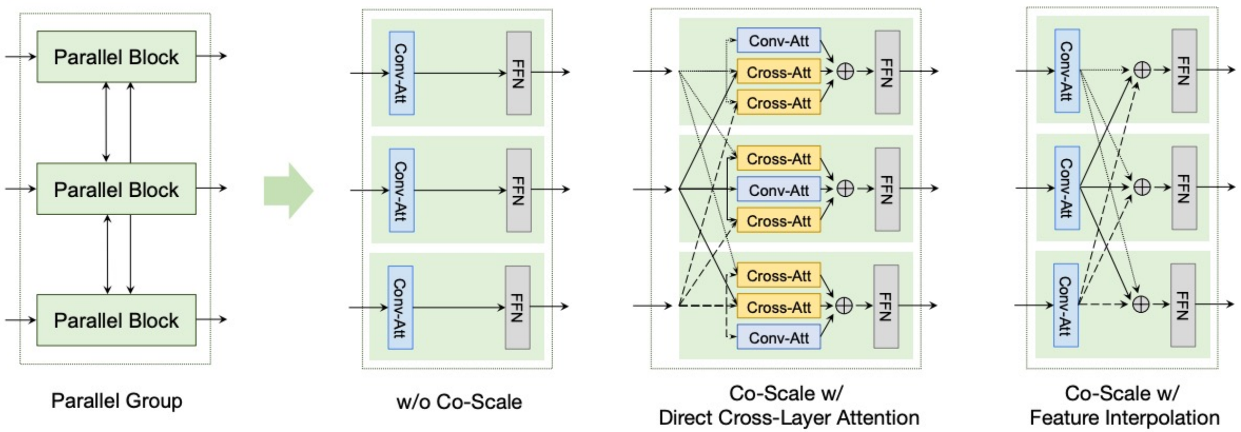
Methods



- **CoaT Serial Block**

- A serial block models image representations in a reduced resolution.
- We first down-sample input feature maps by a certain ratio using a patch embedding layer, and flatten the reduced feature maps into a sequence of tokens.
- Then, we concatenate tokens with an additional class token and apply multiple conv-attentional modules.
- Finally, we separate the class token from image tokens and reshape the image tokens to 2D feature maps for the next serial block.

Methods



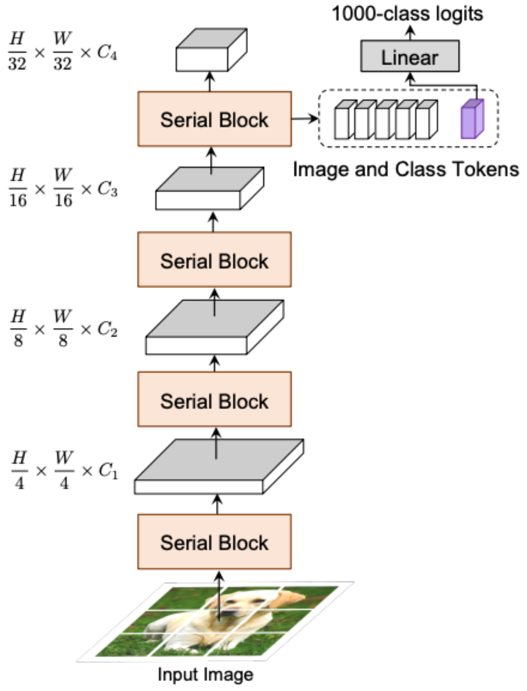
- **CoaT Parallel Block**

- In a parallel group, we have sequences of input features from serial blocks with different scales.
- In direct cross-layer attention, for attention cross different layers, we down-sample or up-sample the key and value vectors to match the resolution of other scales.
- Then, we perform cross-attention.
- Finally, we sum the outputs of conv-attention and cross-attention together and apply a shared feed-forward layer.
- In attention with feature interpolation, first, the input image features from different scales are processed by independent conv-attention modules.
- Then we down-sample or up-sample image features from each scale to match the dimensions of other scales using bilinear interpolation.
- The features belonging to the same scale are summed in the parallel group, and they are further passed into a shared feed-forward layer.

Table of Contents

- Introduction
- Combination of CNN and Self-Attention
- Review Self-Attention in ViT
- Methods
- Model Architecture
- Experiments
- Conclusion

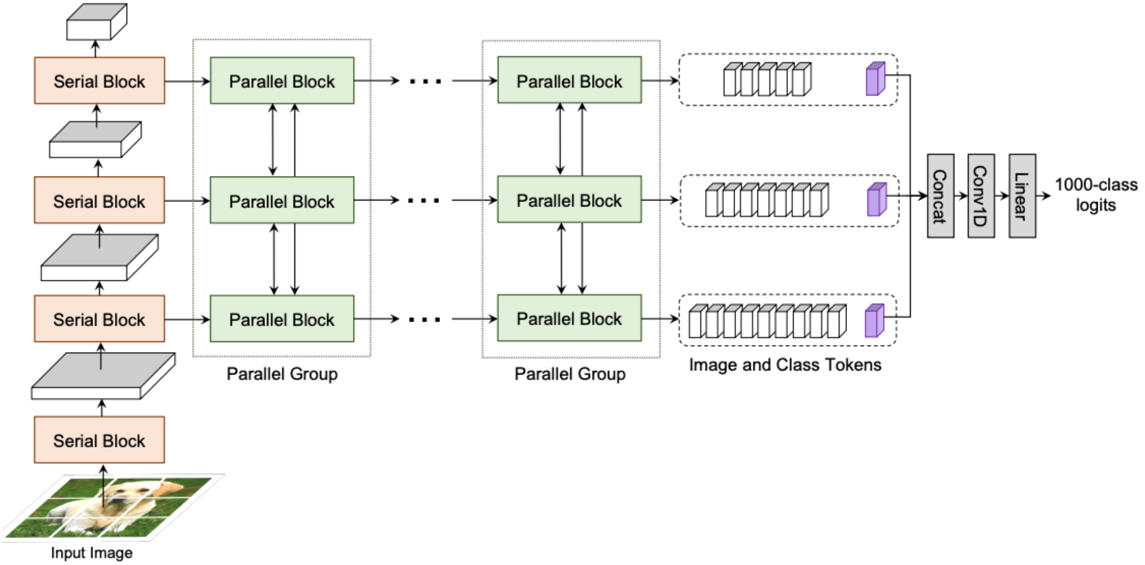
Model Architecture



- **CoaT-Lite**

- This preprocess input images with a series of serial blocks following a fine-to-coarse pyramid structure.
- Given an input image $I \in \mathbb{R}^{H\times W\times C}$, each serial block down-samples the image features into lower resolution, resulting in a sequence of four resolutions: $F_1 \in \mathbb{R}^{\frac{H}{4}\times\frac{W}{4}\times C_1}, F_2 \in \mathbb{R}^{\frac{H}{8}\times\frac{W}{8}\times C_2}, F_3 \in \mathbb{R}^{\frac{H}{16}\times\frac{W}{16}\times C_3}, F_1 \in \mathbb{R}^{\frac{H}{32}\times\frac{W}{32}\times C_4}$.
- In CoaT-Lite, we obtain the class token in the last serial block, and perform classification via a linear projection layer.

Model Architecture



- **CoaT**

- CoaT model consists of both serial and parallel blocks.
- Once we obtain mutl-scale feature maps $\{F_1, F_2, F_3, F_4\}$ from the serial blocks, we pass F_2, F_3, F_4 and corresponding class tokens into the parallel group with three separate parallel blocks.
- To perform classification, we aggregate the class tokens from all three scales.

Model Architecture

Table 1. **Architecture details of CoaT-Lite and CoaT models.** C_i represents the hidden dimension of the attention layers in block i; H_i represents the number of attention heads in the attention layers in block i; R_i represents the expansion ratio for the feed-forward hidden layer dimension between attention layers in block i. Multipliers indicate the number of conv-attentional modules in block i.

Blocks	Output	CoaT-Lite				CoaT		
		Tiny	Mini	Small	Medium	Tiny	Mini	Small
Serial Block (S_1)	56×56	$\begin{bmatrix} C_1 = 64 \\ H_1 = 8 \\ R_1 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 64 \\ H_1 = 8 \\ R_1 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 64 \\ H_1 = 8 \\ R_1 = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} C_1 = 128 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} C_1 = 152 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 152 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 152 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 2$
Serial Block (S_2)	28×28	$\begin{bmatrix} C_2 = 128 \\ H_2 = 8 \\ R_2 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C_2 = 128 \\ H_2 = 8 \\ R_2 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C_2 = 128 \\ H_2 = 8 \\ R_2 = 8 \end{bmatrix} \times 4$	$\begin{bmatrix} C_1 = 256 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} C_2 = 152 \\ H_2 = 8 \\ R_2 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_2 = 216 \\ H_2 = 8 \\ R_2 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 320 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 2$
Serial Block (S_3)	14×14	$\begin{bmatrix} C_3 = 256 \\ H_3 = 8 \\ R_3 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_3 = 320 \\ H_3 = 8 \\ R_3 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_3 = 320 \\ H_3 = 8 \\ R_3 = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} C_1 = 320 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 10$	$\begin{bmatrix} C_3 = 152 \\ H_3 = 8 \\ R_3 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_3 = 216 \\ H_3 = 8 \\ R_3 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 320 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 2$
Serial Block (S_4)	7×7	$\begin{bmatrix} C_4 = 320 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_4 = 512 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_4 = 512 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} C_1 = 512 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 8$	$\begin{bmatrix} C_4 = 152 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_4 = 216 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 320 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 2$
Parallel Group	$\begin{bmatrix} 28 \times 28 \\ 14 \times 14 \\ 7 \times 7 \end{bmatrix}$					$\begin{bmatrix} C_4 = 152 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} C_4 = 216 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} C_1 = 320 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 6$
#Params		5.7M	11M	20M	45M	5.5M	10M	22M

Table of Contents

- Introduction
- Combination of CNN and Self-Attention
- Review Self-Attention in ViT
- Methods
- Model Architecture
- Experiments
- Conclusion

Experiments

Table 2. **CoaT performance on ImageNet-1K validation set.** Our CoaT models consistently outperform other methods while being parameter efficient. ConvNets and ViTNets with similar model size are grouped together for comparison. "#GFLOPs" and Top-1 Acc are measured at input image size. "*" results are adopted from [33].

Arch.	Model	#Params	Input	#GFLOPs	Top-1 Acc.
ConvNets	EfficientNet-B0 [29]	5.3M	224^2	0.4	77.1%
	ShuffleNet [43]	5.4M	224^2	0.5	73.7%
ViTNets	DeiT-Tiny [30]	5.7M	224^2	1.3	72.2%
	CPVT-Tiny [6]	5.7M	224^2	-	73.4%
	CoaT-Lite Tiny (Ours)	5.7M	224^2	1.6	77.5%
	CoaT Tiny (Ours)	5.5M	224^2	4.4	**78.3%**
ConvNets	EfficientNet-B2[29]	9M	260^2	1.0	80.1%
	ResNet-18* [11]	12M	224^2	1.8	69.8%
ViTNets	PVT-Tiny [33]	13M	224^2	1.9	75.1%
	CoaT-Lite Mini (Ours)	11M	224^2	2.0	79.1%
	CoaT Mini (Ours)	10M	224^2	6.8	**81.0%**
ConvNets	EfficientNet-B4 [29]	19M	380^2	4.2	**82.9%**
	ResNet-50* [11]	25M	224^2	4.1	78.5%
	ResNeXt50-32x4d* [36]	25M	224^2	4.3	79.5%
ViTNets	DeiT-Small [30]	22M	224^2	4.6	79.8%
	PVT-Small [33]	24M	224^2	3.8	79.8%
	CPVT-Small [6]	22M	224^2	-	80.5%
	T2T-ViT$_t$-14 [40]	22M	224^2	6.1	81.7%
	Swin-T [17]	29M	224^2	4.5	81.3%
	CoaT-Lite Small (Ours)	20M	224^2	4.0	81.9%
	CoaT Small (Ours)	22M	224^2	12.6	82.1%
ConvNets	EfficientNet-B6 [29]	43M	528^2	19	84.0%
	ResNet-101* [11]	45M	224^2	7.9	79.8%
	ResNeXt101-64x4d* [36]	84M	224^2	15.6	81.5%
ViTNets	PVT-Large [33]	61M	224^2	9.8	81.7%
	T2T-ViT$_t$-24 [40]	64M	224^2	15	82.6%
	DeiT-Base [30]	86M	224^2	17.6	81.8%
	CPVT-Base [6]	86M	224^2	-	82.3%
	Swin-B [17]	88M	224^2	15.4	83.5%
	Swin-B [17]	88M	384^2	47	**84.5%**
	CoaT-Lite Medium (Ours)	45M	224^2	9.8	83.6%
	CoaT-Lite Medium (Ours)	45M	384^2	28.7	**84.5%**

Experiments

Table 3. **Object detection and instance segmentation results based on Mask R-CNN on COCO val2017**. Experiments are performed under the MMDetection framework [4]. "*" results are adopted from Detectron2.

Backbone	#Params (M)	w/ FPN 1×		w/ FPN 3×	
		AP^b	AP^m	AP^b	AP^m
ResNet-18*	31.3	34.2	31.3	36.3	33.2
PVT-Tiny [33]	32.9	36.7	35.1	39.8	37.4
CoaT-Lite Mini (Ours)	30.7	41.4	38.0	42.9	38.9
CoaT Mini (Ours)	30.2	**45.1**	**40.6**	**46.5**	**41.8**
ResNet-50*	44.3	38.6	35.2	41.0	37.2
PVT-Small [33]	44.1	40.4	37.8	43.0	39.9
Swin-T [17]	47.8	43.7	39.8	46.0	41.6
CoaT-Lite Small (Ours)	39.5	45.2	40.7	45.7	41.1
CoaT Small (Ours)	41.6	**46.5**	**41.8**	**49.0**	**43.7**

Table 4. **Object detection and instance segmentation results based on Cascade Mask R-CNN on COCO val2017**. Experiments are performed using the MMDetection framework [4].

Backbone	#Params (M)	w/ FPN 1×		w/ FPN 3×	
		AP^b	AP^m	AP^b	AP^m
Swin-T [17]	85.6	48.1	41.7	50.4	43.7
CoaT-Lite Small (Ours)	77.3	49.1	42.5	48.9	42.6
CoaT Small (Ours)	79.4	**50.4**	**43.5**	**52.2**	**45.1**

Experiments

Table 6. **Effectiveness of position encodings.** All experiments are performed with the CoaT-Lite Tiny architecture. Performance is evaluated on the ImageNet-1K validation set.

Model	CPE	CRPE	Top-1 Acc.
CoaT-Lite Tiny	✗	✗	68.8%
	✗	✓	75.0%
	✓	✗	75.9%
	✓	✓	77.5%

Experiments

Table 7. **Effectiveness of co-scale.** All experiments are performed with the CoaT Tiny architecture. Performance is evaluated on the ImageNet-1K validation set and the COCO val2017 dataset.

Model	#Params	Input	#GFLOPs	Top-1 Acc. @input	AP^b	AP^m
CoaT w/o Co-Scale	5.5M	224^2	4.4	77.8%	41.6	37.9
CoaT w/ Co-Scale						
- Direct Cross-Layer Attention	5.5M	224^2	4.8	77.0%	42.1	38.3
- Attention w/ Feature Interp.	5.5M	224^2	4.4	78.3%	42.5	38.6

Experiments

Table 8. **ImageNet-1K validation set results** compared with the concurrent work Swin Transformer[17]. Computational metrics are measured on a single V100 GPU.

Model	#Params	Input	GFLOPs	FPS	Latency	Mem	Top-1 Acc.	Top-5 Acc.
Swin-T [17]	28M	224^2	4.5	755	16ms	222M	81.2%	95.5%
CoaT-Lite Small (Ours)	20M	224^2	4.0	634	32ms	224M	81.9%	95.6%
CoaT Small (Ours)	22M	224^2	12.6	111	60ms	371M	**82.1%**	**96.1%**
Swin-S [17]	50M	224^2	8.7	437	29ms	372M	83.2%	96.2%
Swin-B [17]	88M	224^2	15.4	278	30ms	579M	83.5%	96.5%
CoaT-Lite Medium (Ours)	45M	224^2	9.8	319	52ms	429M	**83.6%**	**96.7%**
Swin-B [17]	88M	384^2	47.1	85	33ms	1250M	**84.5%**	97.0%
CoaT-Lite Medium (Ours)	45M	384^2	28.7	97	56ms	937M	**84.5%**	**97.1%**

Table of Contents

- Introduction
- Combination of CNN and Self-Attention
- Review Self-Attention in ViT
- Methods
- Model Architecture
- Experiments
- Conclusion

Conclusion

- In this paper, we present a Transformer based image classifier, CoaT.
- This models attain strong classification results on ImageNet.
- And their applicability to downstream tasks has been demonstrated for object detection and instance segmentation.